AMD

SYSCALL and SYSRET Instruction Specification

Application Note

Publication # **21086** Rev: **C** Amendment/**0**
Issue Date: **May 1998**

Contents

Revision History

Date	Rev	Description
Sept 1997	B	Initial published release.
May 1998	C	Revised code sample on testing for SYSCALL/SYSRET support on page 2.

Application Note

SYSCALL and SYSRET Instruction Specification

Audience

The SYSCALL and SYSRET instruction specification application note is intended for operating system designers.

Overview

SYSCALL and SYSRET are instructions used for low-latency system calls and returns in operating systems with a flat memory model and no segmentation. These instructions have been optimized by reducing the number of checks and memory references that are normally made so that a call or return takes less than one-fourth the number of internal clock cycles when compared to the current CALL/RET instruction method.

Feature Detection

To use the SYSCALL/SYSRET instructions, the programmer must determine if the processor supports them. The CPUID instruction gives programmers the ability to determine the presence of these instructions on the processor. Software must

first test to see if the CPUID instruction is supported. For a detailed description of the CPUID instruction, see the *AMD Processor Recognition Application Note*, order# 20734.

The presence of the CPUID instruction is indicated by the ID bit (21) in the EFLAGS register. If this bit is writeable, the CPUID instruction is supported. The following code sample shows how to test for the presence of the CPUID instruction.

```
pushfd                  ; save EFLAGS
pop   eax               ; store EFLAGS in EAX
mov   ebx, eax          ; save in EBX for later testing
xor   eax, 00200000h    ; toggle bit 21
push  eax               ; put to stack
popfd                   ; save changed EAX to EFLAGS
pushfd                  ; push EFLAGS to TOS
pop   eax               ; store EFLAGS in EAX
cmp   eax, ebx          ; see if bit 21 has changed
jz    NO_CPUID          ; if no change, no CPUID
```

If the processor supports the CPUID instruction, the programmer must execute the standard function, EAX=0. The CPUID function returns a 12-character string that identifies the processor's vendor. For AMD processors, standard function 0 returns a vendor string of "AuthenticAMD". This string requires the software to follow the AMD definitions for subsequent CPUID functions and the values returned for those functions.

The next step for the programmer is to determine if the SYSCALL/SYSRET instructions are supported. Extended function 8000_0001h of the CPUID instruction provides this information. Function 8000_0001h (EAX=8000_0001h) of the AMD CPUID instruction returns the feature bits in the EDX register. If bit 11 in the EDX register is set to 1, SYSCALL/SYSRET is supported. The following code sample shows how to test for SYSCALL/SYSRET support.

```
mov   eax, 80000001h    ; setup function 8000_0001h
CPUID                   ; call the function
test  edx, 800h         ; test bit 11
jnz   YES_SYSCALL/SYSRET
```

Background

The x86 operating system designers use both segmentation and paging to implement various memory models in their designs. Memory can be protected or unprotected, and segmentation and paging can be implemented with attributes such as supervisor, user, read, write, and execute.

Segment descriptors provide the necessary memory protection and privilege checking of segment accesses. By setting the fields within the segment descriptors appropriately, operating systems can enforce access restrictions as needed. One disadvantage of segment-based protection and privilege checking is the overhead associated with the loading of new segments (and their descriptors) into segment registers. Even with pure 32-bit code, this overhead still occurs when switching between ring or privilege levels, and Code Segment (CS) and Stack Segment (SS) are reloaded with different segment descriptors.

To initiate a call to the operating system, an application transfers control to the OS through gate descriptors (task, interrupt, trap, or call gates). Control transfer is done by using either a CALL instruction or a software interrupt. (The Windows® 95 operating system uses CALL gates while Windows NT® uses software interrupts.) Setting up these control gates (as well as the later return via a RET or IRET instruction) is slowed down by the segmentation-related overhead. For example, a CALL to a call gate must initiate the following checks to ensure protection and stability:

- The CS selector must not be a NULL selector. If the selector is NULL, a General Protection Fault occurs.
- The Type field must define the selected descriptor as a code segment descriptor. If the descriptor is not a code segment descriptor, a General Protection Fault occurs.
- The CS selector must be able to index within the limits of the descriptor table. If the index is not within these limits, a General Protection Fault occurs.
- The Descriptor Privilege Level of the call gate must be greater than or equal to the Requestor Privilege level. If it is not, a General Protection Fault occurs.

- The Descriptor Privilege Level of the call gate must be greater than or equal to the Current Privilege Level. If it is not, a General Protection Fault occurs.
- The Stack Segment selector must not be a NULL selector. If the selector is NULL, an Invalid Task State Segment Exception occurs.
- The Stack Segment selector must be able to index within the limits of the descriptor table. If the index is not within these limits, an Invalid Task State Segment Exception occurs.
- The Descriptor Privilege Level of the Stack Segment must be equal to the Descriptor Privilege Level of the Code Segment. If it is not, an Invalid Task State Segment Exception occurs.
- The Requestor Privilege Level of the Stack Segment must be equal to the Descriptor Privilege Level of the Code Segment. If it is not, an Invalid Task State Segment Exception occurs.
- The stack created by the call must be large enough to accommodate a predetermined number of bytes. If not, a Stack Fault Exception occurs.
- The Stack Segment must be present. If not, a Stack Fault Exception occurs.
- The descriptor of the Stack Segment must define the data segment as writeable. If not, an Invalid Task State Segment Exception occurs.

This list is an example of some of the checks that must occur. In addition, there are other checks that involve the Task State Segment.

New Method

The SYSCALL and SYSRET instructions are ideally suited for an operating system with a flat memory model (such as Windows NT) where segmentation is disregarded by making all segments 4 Gbytes (base=0 and limit =4 Gbytes).

By making the assumption that the operating system design is flat, calls and returns can be greatly simplified. Such an assumption also allows for greater security and reliability by

forcing all applications to use a common and consistent view of the memory model. This simplification comes from eliminating unneeded checks as well as loading pre-determined values into the on-chip Code Segment and Stack Segment descriptor caches.

It is assumed that the base, limit, and attributes of the Code Segment will remain the same for all processes and for the operating system, and that only the current privilege level for the selector of the calling process should be changed from a current privilege level of three to a new privilege level of zero. It is also assumed (but not checked) that the privilege level of the selectors for the CALL and RET target CS segments are set to zero and three, respectively (for SYSCALL—code segment selector requested privilege level=0 and for SYSRET—code segment selector requested privilege level=3).

The SYSCALL and SYSRET instructions should be used in environments where neither the operating system nor the applications alter the Code Segment or Stack Segment in a manner different than what is assumed by the instructions. The SYSCALL instruction sets the base, limit, and attributes of both CS and SS to fixed values on entry. The SYSRET instruction sets CS to the same fixed values upon exiting. In a flat memory model, only the privilege level of the segments need to change. No memory accesses to descriptors are required. It is the responsibility of the operating system to keep the descriptors in memory that correspond to the CS and SS selectors loaded by the SYSCALL and SYSRET instructions consistent with the segment base, limit, and attribute values forced by these instructions.

SYSCALL/SYSRET Target Address Register (STAR–MSR C000_0081h)

The SYSCALL/SYSRET Target Address Register (STAR) contains the target EIP address used by the SYSCALL instruction and the 16-bit code and stack segment selector bases used by the SYSCALL and SYSRET instructions. Figure 1 shows the format of the STAR register, and Table 1 defines the function of each bit of the STAR register.

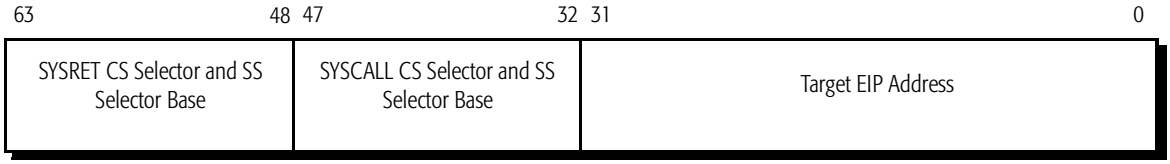

Figure 1. SYSCALL/SYSRET Target Address Register (STAR)

Table 1. SYSCALL/SYSRET Target Address Register (STAR) Definition

Bit	Description	R/W	Function
63–48	SYSRET CS and SS Selector Base	R/W	During the SYSRET instruction, this field is copied into the CS register and the contents of this field, plus 1000b, are copied into the SS register.
47–32	SYSCALL CS and SS Selector Base	R/W	During the SYSCALL instruction, this field is copied into the CS register and the contents of this field, plus 1000b, are copied into the SS register.
31–0	Target EIP Address	R/W	During the SYSCALL instruction, this address is copied into the EIP and points to the new starting address

SYSCALL

mnemonic	*opcode*	*description*
SYSCALL	0F 05h	Call operating system

Privilege: None
Registers Affected: ECX, EIP, CS, SS
Flags Affected: IF, VM
Machine State Affected: CPL, CS (base, limit, attr), SS (base, limit, attr)
Exceptions Generated:

Exception	Real	Virtual 8086	Protected	Description
Invalid opcode (6)	X	X	X	The System Call Extension bit (SCE) of the Extended Feature Enable Register (EFER) is set to 0. (The EFER register is MSR C000_0080h.)

The SYSCALL instruction provides a fast method for transferring control to a fixed entry point in an operating system.

The EIP register is copied into the ECX register. Bits [31–0] of the 64-bit SYSCALL/SYSRET Target Address Register (STAR) are copied into the EIP register. (The STAR register is Model-Specific Register C000_0081h.)

The IF and VM flags are set to 0 in order to disable interrupts and force the processor out of Virtual-8086 mode.

New selectors are loaded without any checking performed as follows:

- Bits [47–32] of the STAR register specify the selector that is copied into the CS register.
- Bits [47–32]+1000b of the STAR register specify the selector that is copied into the SS register.

  Note: *This effectively increments the index field of the CS selector such that the resultant SS selector points to the next descriptor in a descriptor table, after the CS descriptor.*

The CS and SS registers should not be modified by the operating system between the execution of the SYSCALL instruction and its corresponding SYSRET instruction.

The CPL is set to 0 regardless of the value of bits [33–32] of the STAR register. There are no permission checks based on the CPL, Real mode, or Virtual-8086 mode.

The following descriptors are loaded to specify fixed, 4-Gbyte flat segments as follows:

- The CS_base and the SS_base are both set to zero.
- The CS_limit and the SS_limit are both set to 4 Gbyte.
- The CS segment attributes are set to Read-only.
- The SS segment attributes are set to Read/Write and Expand-Up.

The operating system must ensure that the descriptors corresponding to the selectors in the STAR register are consistent with the corresponding on-chip values (such as, base, limit, and attributes) that are set by the SYSCALL and SYSRET instructions.

Related Instructions See the SYSRET instruction.

SYSRET

mnemonic	*opcode*	*description*
SYSRET	0F 07h	Return from operating system

Privilege: None
Registers Affected: EIP, SS, CS
Flags Affected: IF
Machine State Affected: CPL, CS (base, limit, attr)
Exceptions Generated:

Exception	Real	Virtual 8086	Protected	Description
Invalid opcode (6)	X	X	X	The System Call Extension bit (SCE) of the Extended Feature Enable Register (EFER) is set to 0. (The EFER register is MSR C000_0080h.)
General protection (13)	X	X	X	The CPL is not equal to 0.

The SYSRET instruction is the return instruction used in conjunction with the SYSCALL instruction to provide fast entry/exit to an operating system.

The ECX register, which points to the next sequential instruction after the corresponding SYSCALL instruction, is copied into the EIP register.

The IF flag is set to 1 in order to enable interrupts.

New selectors are loaded without any checking performed as follows:

- Bits [63–48] of the STAR register specify the selector that is copied into the CS register.
- Bits [63–48]+1000b of the STAR register specify the selector that is copied into the SS register.

  Note: *This effectively increments the index field of the CS selector such that the resultant SS selector points to the next descriptor in a descriptor table, after the CS descriptor.*
- Bits [1–0] of the SS register are set to 11b (RPL of 3) regardless of the value of bits [49–48] of the STAR register.

The CS and SS registers must not be modified by the operating system between the execution of the SYSCALL instruction and its corresponding SYSRET instruction.

The CPL is set to 3 regardless of the value of bits [49–48] of the STAR register. If the CPL is not equal to 0 when the SYSRET instruction is executed, a general protection fault exception is generated with an error code of 0.

A new descriptor is loaded for CS to specify a fixed 4-Gbyte flat segment as follows:

- The CS_base is set to zero.
- The CS_limit is set to 4 Gbyte.
- The CS segment attributes are set to Read-only.

The operating system must ensure that the descriptors corresponding to the selectors in the STAR register are consistent with the corresponding on-chip values (such as, base, limit, and attributes) that are set by the SYSCALL and SYSRET instructions.

Related Instructions See the SYSCALL instruction.